

by
Andy Phillips

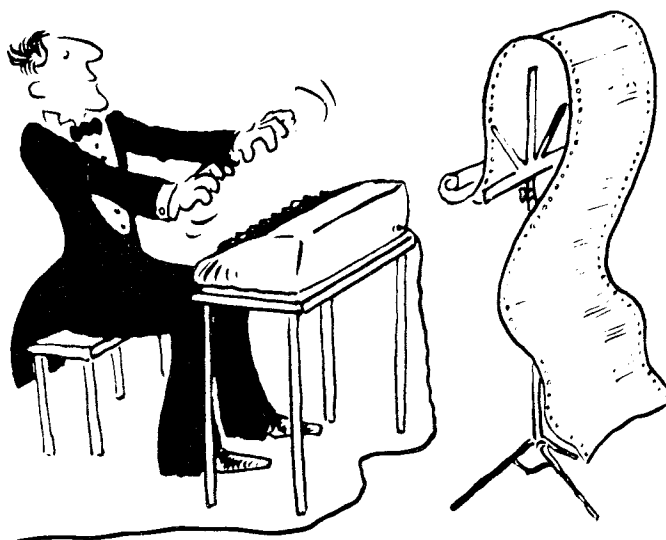
The Hamilton Board of Education
1983

I N T E R A C T I V E P R O G R A M M I N G

TABLE OF CONTENTS









<u>Topic #</u>	<u>Title</u>	<u>Page</u>
1	THE KEYBOARD	1
2	ONE LINE PROGRAMMES	6
3	COMMODORE BASIC	10
4	COMPUTER STORAGE OF DATA	12
5	BRANCHING	17
6	DECISION MAKING	20
7	LOOPS	22
8	INTERACTIVE BASIC	26
9	DATA STATEMENTS	30
10	SOME BUILT-IN FUNCTIONS	33
11	SUBROUTINES	39
12	ARRAYS	44
13	THE CASSETTE RECORDER	46
14	THE PRINTER	49
--	EXTRA NOTES:	
	A. PET BASIC RESERVED WORDS	51
	B. ALTERNATE CHARACTER SET	51
	C. USER DEFINED FUNCTIONS	52
	D. ENHANCED CHARACTER PRINTOUT	52

THE KEYBOARD



TOPIC #1: THE KEYBOARD

- ON/OFF 1. Find the black rocker switch at the back of the PET and press it to turn on the PET.
2. The first line of the display on the screen indicates the language in which the PET can be programmed. What is the name of this language?
- BYTES 3. The second line of the display indicates the number of storage locations available in which to store characters. Each BYTE can be used to store a single character in memory. According to the display, how many characters can the PET store?
- K 4. A computer that can store approximately 8000 characters is said to have an 8K (K = 1024) memory. How would you describe the size of the memory of your PET?
- READY 5. The third line of the display contains the word READY. This word will appear whenever the PET is able to accept your instructions.
- CURSOR 6. Under the word READY appears a small flashing square. This is the CURSOR. It is used to indicate the position on the screen of the next character to be displayed.
7. Using the black rocker switch, turn the PET off, and then back on again. The screen should momentarily fill with a random display. This is normal: You can ignore it.
8. The screen should again show the initial display mentioned above. What is the name given to the little flashing square that appears below the word READY?
- RETURN 9. Find the RETURN key on the keyboard and press it. What happened to the cursor?
10. Whenever you have finished typing a line of instructions you must press the RETURN key to have the PET execute that line. If, when you are typing in instructions, you press the wrong key, or leave out a letter, press the RETURN key and type the line over. If you get an ERROR message just ignore it for now: It will not affect the next line that you type.
11. Type in your name and press the RETURN key. What error message did you get?
12. Since you are not writing programmes at present, you can ignore any error messages for the time being.
13. Remember, if the word READY does not appear on the last printed line on the screen, you can always press the RETURN key to have it re-appear. Try this by pressing the RETURN key once more.

- OFF/RVS 14. Find the OFF/RVS key on the keyboard and press it. Type your first name and press the space bar once or twice. Now hold one of the SHIFT keys down and press the OFF/RVS key once more. This turns the reverse field off. Test it by typing your last name. Finish by pressing the RETURN key.
- GRAPHICS 15. On the front face of most keys, you will find a different graphic symbol. These may be used to construct graphic displays on the PET. To get the PET to print these displays, it is necessary to hold one of the SHIFT keys down while you press the desired key at the same time. If you are going to press more than one of these keys, you can hold the SHIFT key down by pressing the SHIFT/LOCK key. (Press it again to release it).
- SHIFT/LOCK 16. Experiment by pressing a number of these keys in turn. Use the OFF/RVS key to obtain more interesting designs. Remember to press the RETURN key after you are finished typing on each line.
- SCROLLING 17. You should have noticed that when you got to the bottom of the screen, the entire display popped up one line to allow you to print another line on the screen. This is known as SCROLLING and it is automatic on the PET.
- CLR/HOME 18. Hold the SHIFT key down, then locate and press the CLR/HOME key. What happened to the screen?
What happened to the cursor?
19. The CLR/HOME key can be used whenever you wish to create a blank screen. To leave the display as it is and yet move the cursor back to the initial print position, you just press the CLR/HOME key without holding down the shift key.
- INST/DEL 20. Once again type your name, but do not press the RETURN key just yet. Now press the INST/DEL (delete) key a few times. What does the INST/DEL key do?
21. Finish retyping your name and then press the RETURN key. Now type the word MISTEAK and use the INST/DEL key to correct it to MISTAKE before pressing the RETURN key.
- CURSOR CONTROLS 22. Type the phrase I WONDER WHERE THE CURSOR IS? Do not press the RETURN key yet. Now hold down the SHIFT key and locate and press the /CRSR/ key a few times. What happened to the cursor?
23. Now release the SHIFT key and press the /CRSR/ key again a few times. What happened to the cursor?
24. Locate and press the /CRSR/ key a number of times. What happened to the cursor?
25. Now hold the SHIFT key down and press the /CRSR/ key once or twice. What happened?

26. Use the 'cursor control' keys mentioned above to locate the cursor at the N in the word WONDER. (If the phrase has disappeared, retype it). Now press the INST/DEL key. What letter vanished? What happened to all the letters to the right of the O?

INST/DEL
































27. Now hold the SHIFT key down and press the INST/DEL key again. What happened this time?
28. Move the cursor, using the control keys, onto the newly created blank space, and type the letter A. Then press the RETURN key.
29. The INST/DEL key can be used to make corrections. Practice this technique by typing in various phrases with letters missing or words misspelled, and correct them.
30. Notice that you do not get an error message when working strictly with graphics. Try to produce each of the following designs (and some of your own) on the screen. Clear the screen after each design.
- a) A solid border going right around the screen (Hint: Use the OFF/RVS key and the SPACE BAR).
 - b) An isosceles triangle, the base of which is 10 characters wide (Hint: Use the graphic symbols on the M and N keys for the slant sides of the triangle).
 - c) A small truck, motorcycle, or car.
 - d) A space vehicle similar to those seen in the "Space Invaders" video game.
 - e) A checkerboard pattern, each square of which is 3 characters wide, and three lines deep.
 - f) A playing card, such as the "3 of hearts".

*HINT: When using the PET to create graphic displays you should use the cursor control keys to move from line to line. If you use the RETURN key you may get an error message which would interfere with your display. Pressing RETURN while the shift is down will eliminate this problem.

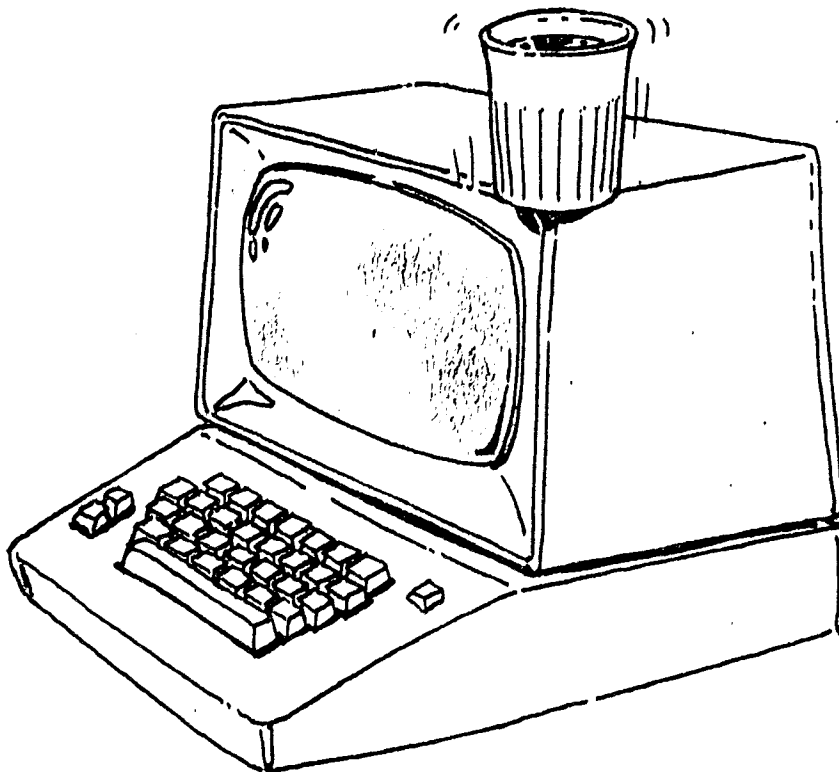
1. What language do we use to programme the PET?
2. How many characters can be stored in a single byte?
3. What size memory does your PET have?
4. What is the name of the little flashing square which indicates the next print position?
5. What key must be pressed after each line is typed so that it will be executed by the PET?
6. What key is used to reverse the field of each character, and how is this reversing turned off?
7. What word is used to describe the upward movement of the display when the cursor reaches the bottom of the screen?
8. If we want to get the cursor back to the top left print position what key do we press?
9. How can the entire display be wiped out?
10. What key is used to eliminate the character to the left of the cursor?
11. On which side of the cursor are blanks inserted by the INST/DEL key?



EQUIVALENT FUNCTION CHART

FUNCTION	COMMODORE		APPLE
	SCREEN SYMBOL	KEYS USED	
CLEAR SCREEN (cursor home)	"  "	 Hold Down  Strike	Type . . . HOME
DELETE (erases errors)	"  "		Use ← or → to get over the error; retype; Use → to get off printed items; RETURN key
CURSOR MOVEMENT	Down 		Press ESC key . . . Then one of these keys . . .    
	Up 	 	
	Right 		
	Left 	 	
REVERSE VIDEO	On 		Type . . . INVERSE e.g.: 10 INVERSE 20 PRINT "HELLO"
	Off 	 	Type . . . NORMAL e.g.: 30 NORMAL
FLASHING LINE			Type . . . FLASH e.g.: 10 FLASH 20 PRINT "HELLO"
INSERTING CHARACTERS	"  "	 	• Not Available • Retype the Line
EXPONENTIAL			

ONE LINE PROGRAMMES



"DON'T BRING DRINKS OR FOOD
INTO THE COMPUTER ROOM."

TOPIC #2: ONE LINE PROGRAMMES

PRINT

1. Turn on the PET and clear the screen. Now type in the following phrase:

PRINT "MY VERY FIRST PROGRAMME"

You will notice that the double quotes appear on a single key. Now press the RETURN key. What is printed out on the next line below the one you typed?

2. Any expression that you type, within double quotes, after the command PRINT will be printed out when you press the RETURN key. Experiment by typing each of the following lines (Do not forget to press the RETURN key after each line).

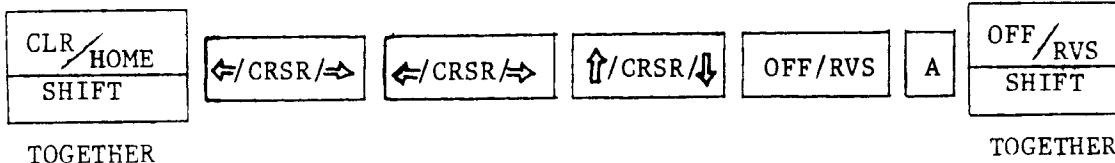
a) PRINT "1+2=3"

b) PRINT "4♦'s + 3♦'s = 7♦'s"

c) PRINT "HERE WE GO!"

SPECIAL
SYMBOLS

3. Certain keys, such as the OFF/RVS key and the cursor control, behave differently when they follow an odd number of double quotation marks. Type in PRINT then a double quotation mark, then press the following keys.



Now finish the line by typing a double quote. Make note of the symbols used to represent each of these keys.

CLR/HOME and SHIFT _____

⇐/CRSR/⇒ _____

↑/CRSR/↓ _____

OFF/RVS _____

OFF/RVS and SHIFT _____

4. Now press the RETURN key and notice the results:

1) The SHIFTED CLR/HOME key clears the screen.

2) The cursor is moved to the right 2 spaces by the two
⇐/CRSR/⇒ keys.

3) The cursor is moved down one line by the ↑/CRSR/↓ key.

4) A reversed A is printed at this location.

5. Make up your own PRINT statement to determine the symbols for CLR/HOME, \leftarrow /CRSR/ \rightarrow and SHIFT, \uparrow /CRSR/ \downarrow and SHIFT, and INST/DEL and SHIFT together.

CLR/HOME _____

\leftarrow /CRSR/ \rightarrow and SHIFT _____

\uparrow /CRSR/ \downarrow and SHIFT _____

INST/DEL and SHIFT _____

- PUNCTUATION 6. More than one expression can be printed by a single PRINT command. Commas and semicolons are used to separate these expressions. Type in the following lines and press the RETURN key after each.

a) PRINT "A"; "B"; "C"; "D"

b) PRINT "A", "B", "C", "D"

c) PRINT "A"; "B", "C"; "D"

What happens when a ';' is used?

What happens when a ',' is used?

- OPERATIONS 7. Mathematical expressions are calculated and their answers printed when they follow the PRINT command. Type in each of the following lines and press the RETURN key after each.

a) PRINT 2-3

b) PRINT 5*7

c) PRINT 4 \uparrow 2

d) PRINT 3-1*3+14/2

e) PRINT 3*2 \uparrow 2

8. What is the symbol used for multiplication in BASIC?
What symbol is used for raising numbers to exponents?
Calculate (d) above yourself using the order of operations that you were taught in math. Does the PET follow this order of operations?

- SPACING 9. Type in the following line:

PRINT 5+2;3-7;4 \uparrow 2

and press the RETURN key. Notice the spacing of the answers. How many blanks appear to the left of a positive number?

How many blanks appear to the left of a negative number?

When a negative number follows another number, how many spaces are inserted between them?

When a positive number follows another number, how many spaces are inserted between them?

10. This can be summarized by the following rule: When a number is printed a space is provided to the left of the number for its sign, (If it is positive, this space is left blank) and to the right of the number to separate it from other numbers that may be printed on the same line.

11. Type in the following lines:

a) PRINT "2+3=";2+3

b) ? "FOOT + BALL=";"FOOT"+"BALL"

Notice that the expressions found completely inside double quotation marks are printed out as they appear, but those without quotes are calculated and the final answer is printed. Notice that the ? key can be used instead of the word PRINT, and that expressions in double quotes can be added together to form one word. This is known as concatenation.

REVIEW QUESTIONS

1. Prepare a list of all the special symbols used inside double quotation marks to represent the special keys such as the OFF/RVS key and the cursor control keys.
2. When a comma is used to separate expressions in a PRINT statement, each successive expression is printed starting at columns 1, 11, 21, and 31. If a semicolon is used, where are successive expressions printed?
3. What is the order of operations followed by the PET?
4. What output would you expect from the following BASIC statement?

? 3-5;4↑3;12/6+2;3*2↑3

Include the spacing you would expect to see.

5. What word is used to describe the 'adding' together of two expressions in double quotation marks?



EXERCISE

- (6) Type in the following commands and take note of what appears on the screen:

- (A) PRINT "HELLO"
- (B) PRINT 'HELLO'
- (C) PRINT "HELLO
- (D) PRNT "HELLO"
- (E) PRINT "!!?#"

(Remember to press RETURN
at end of each line.)

- (7) Type in the following commands and then use the DEL key to make the suggested changes:

- (A) PRINT "HELLO THEIR" → PRINT "HELLO THERE"
- (B) PRINT "POOS" → PRINT "OOPS"
- (C) PINT 3+9 → PRINT 3+9

- (8) The GRAPHICS characters are obtained by depressing the SHIFT key together with one of the keyboard keys. Experiment with this set by making up various patterns and printing them out.

Example: PRINT " ♥ ♦ "

- (9) Determine the value of these expressions. Then check if your answer agrees with the computer's evaluation.

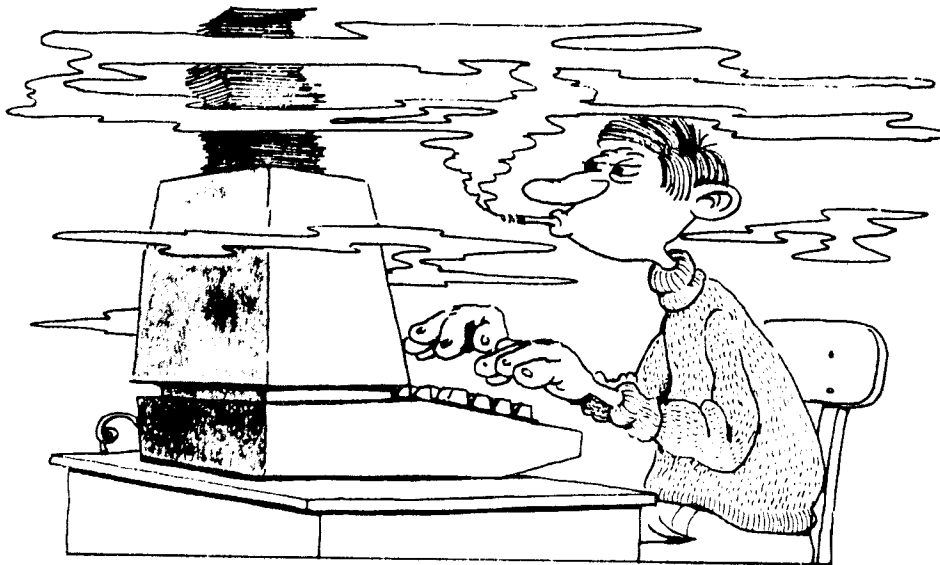
- (A) PRINT 3 - 2 + 2 + (5 - 2) / 3 + 1
- (B) PRINT 10 - 5 + 2 * 2 / 2 + 8 * 3

- (10) This formula is used to calculate compound interest:

$$\text{Final Amount} = P(1 + i)^n$$

Using $P = 1000$, $i = .18$ and $n = 5$, write the PRINT command that will evaluate the data and print the answer. Did you get 2287.75776?

COMMODORE BASIC



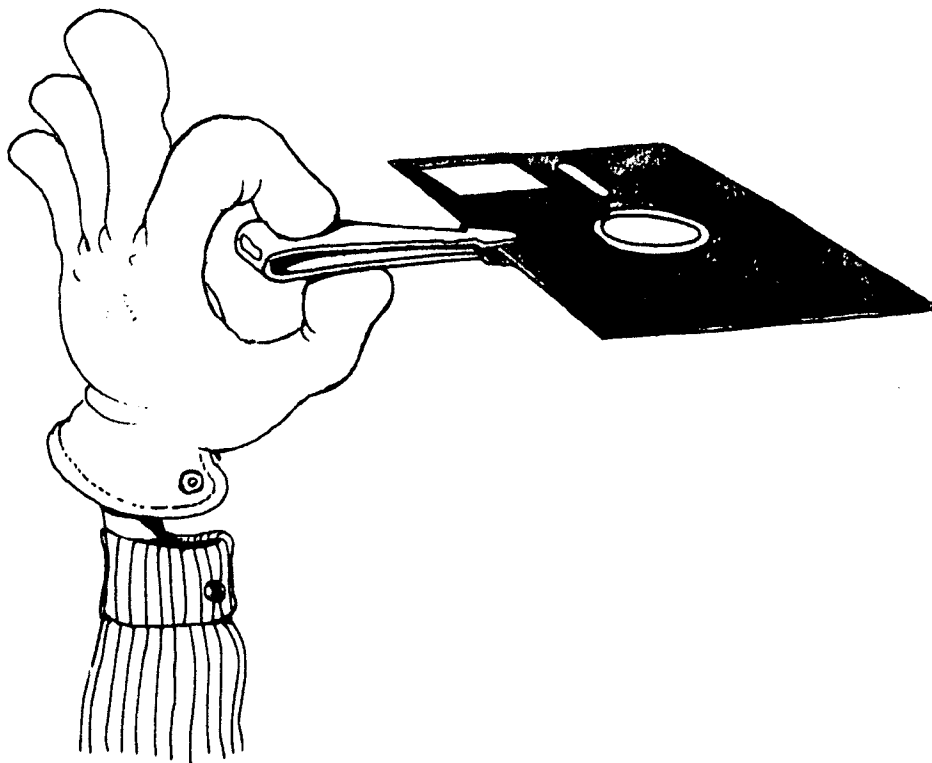
"SMOKE PARTICLES WILL RUIN DISKETTES."

TOPIC #3: COMMODORE BASIC

- NEW 1. Turn on the PET and clear the screen. Type in the word NEW and press the RETURN key. Every time you wish to type in a BASIC programme, you should begin by clearing the PET's memory using the NEW command.
- LINE 2. Every line in BASIC should begin with a line number. This can be any number from 1 to 63999. We usually use 10, 20, 30, and so on in case we need to insert lines later on. Type in the following lines (remember to press the RETURN key after each line).
- ```
10 REM PROGRAMME ONE
20 LET A=2
30 B=3
40 PRINT A+B
50 END
```
3. What you have typed in is a simple programme which takes two numbers, adds them, and prints out the answer.
- REM            4. Line number 10 is a REM (remark) statement. It allows you to comment on parts of your programme. You can type almost anything after a REM. The computer will ignore the rest of the statement. Long, complicated programmes should have many REM statements, each describing a different aspect of the programme.
- LET            5. Numbers can be stored in the PET's memory using a LET (assignment) statement. Line 20 contains an assignment statement which causes the value 2 to be stored in the variable (memory location) A. The LET may be omitted. Line 30 contains an assignment statement which causes the value 3 to be stored in the variable (memory location) B.
- NUMERIC  
VARIABLES      6. A BASIC variable name can contain up to 255 characters. The first must be a letter from A to Z. The second can be either a letter from A to Z or a digit from 0 to 9. The other characters can be letters or digits also, but only the first two characters refer to the actual storage location. Thus AB, ABC, AB9, and AB9C5 all refer to the same storage location AB.
- RESERVED  
WORDS          7. You must take care, when creating variable names, that you do not include a RESERVED WORD somewhere in the name. For example TOTAL contains the RESERVED WORD 'TO' and would result in an error message if it were used as a variable. See the table of RESERVED WORDS which follow these notes, and avoid using them in your variable names.
- RUN            8. To 'run' this programme, type the command RUN and press the RETURN key. The number 5 should appear below the word RUN on the screen.
9. This programme is stored in the PET's memory. To run it again just type RUN and press the RETURN key.

- LIST
10. Now clear the screen. To see if the PET still remembers your programme, list it by typing LIST and then pressing the RETURN key. Notice that the entire programme is listed on the screen. The PET will space out your statements using its own rules so that the new listing may appear a little more spread out than before.
  11. Now run it again. You should see that it still works, giving the answer 5.
- EDIT
12. Next EDIT your programme using the cursor control keys. Move the cursor to line 20 and delete the word LET. You must press the RETURN key after editing a line! Set the cursor on line 40 and change the addition sign to a subtraction sign.
  13. Drop the cursor to the next available blank line and type in LIST, then press the RETURN key. Compare the new programme with the original. If one of the changes you tried to make did not occur, you probably forgot to press the RETURN key after making the change.
  14. Now run the programme again. This time you should get -1 as an answer.
- END
15. Line number 50 may be omitted. The statement END just tells the PET not to execute any more statements after it has executed line 50. It is a good programming technique to use the END statement to signify the end of your programme.

COMPUTER STORAGE OF DATA



" HANDLE DISKETTES WITH CARE "



#### TOPIC #4: COMPUTER STORAGE OF DATA

A computer can store information in its memory. The information can be either . . .

- (a) Numeric - Numbers which can be used for arithmetic purposes.
- (b) Literal (Strings) - Letters, characters, symbols, or numbers which are used for such things as messages, titles, addresses, etc.

The computer stores everything as a number code, though. Thus, the BASIC language must be able to convey to the computer how to interpret the number codes in its memory: as a number, or as a code for a letter or symbol.

A simplistic model for the computer's memory (as used by a BASIC program) follows:

DATA can be stored in memory locations.

Each memory location is like a mail box in a post office.

Every memory location has a unique address (like a mail box).

|   |    |    |    |    |     |
|---|----|----|----|----|-----|
| A | A0 | A1 | A2 | A3 | ... |
| B | B0 | B1 | B2 | B3 | ... |
| C | C0 | C1 | C2 | C3 | ... |

#### Storage Locations for Numeric Data:

1. Numbers can be stored in any memory location with an address such as:

A, A0, A1, A2, . . . , A9, AA, AB, AC, . . . , AZ  
B, B0, B1, B2, . . . , B9, BA, BB, BC, . . . , BZ  
.  
.  
.  
Z, Z0, Z1, Z2, . . . , Z9, ZA, ZB, ZC, . . . , ZZ

2. To store a number in any particular memory location, an assignment statement can be used.

Example:    10   LET A = 5                      OR        10   A = 5  
             20   LET M3 = 10                    20   M3 = 10

the word LET  
is optional

The resultant contents of memory would be:

|        |         |     |          |
|--------|---------|-----|----------|
| A<br>5 | A0<br>0 | ... | 0        |
| B<br>0 | B0<br>0 | ... | M3<br>10 |

\* Note: When your program begins, the computer sets all storage locations equal to ZERO. These values are changed only by commands in your program which store new values. (A new value replaces an old one.)

3. An assignment statement can also be used to perform an arithmetic calculation first, then to store the answer.

Example: 50 AN = A + B

Take the contents of memory locations A and B, add them together, and store the answer in memory location AN.

i.e.:

|         |         |         |
|---------|---------|---------|
| A<br>17 | B<br>45 | AN<br>0 |
|---------|---------|---------|

before execution

|         |         |          |
|---------|---------|----------|
| A<br>17 | B<br>45 | AN<br>62 |
|---------|---------|----------|

after execution

### EXERCISE

- (1) Determine the contents of memory and the output produced by this program:

```

10 H = 40
20 R = 5.95
30 G = H * R
40 PRINT "GROSS PAY"
50 PRINT G
60 END

```

memory

|   |   |   |
|---|---|---|
| H | R | G |
|---|---|---|

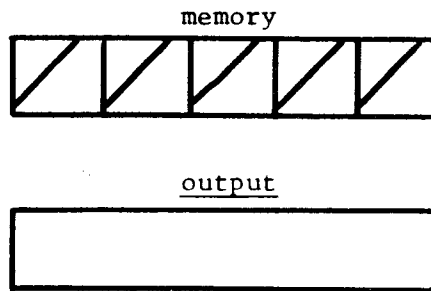
output

|  |
|--|
|  |
|--|

- (2) Modify the program in (1) so that a deduction (D) of 25.45 would be incorporated, and the net pay would be printed out below the gross pay. Use a heading.

- (3) Work through the following program, determining the memory contents and output:

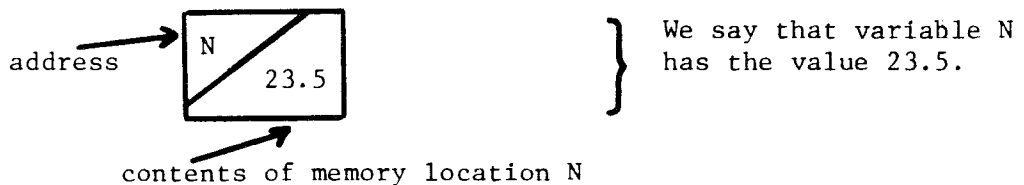
```
10 M1 = 50
20 M2 = 65
30 M3 = 42
40 T = M1 + M2 + M3
50 A = T / 3
60 PRINT T
70 PRINT A
80 END
```



- (4) Modify the program in (3) so that headings precede each of the output values.

Variable Names:

1. The addresses used for memory locations are called variable names.



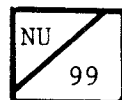
The term "variable" is used because the value (contents) can be changed during the program. Remember that the last value assigned to a variable replaces any former value it may have had.

2. Variable names must start with a letter. They can (for the PET) be any length. However, the computer only uses the first two characters for the address.

Example: • program includes:

```
70 NUMBER = 99
```

• computer stores the value 99 in location NU.



The longer word may make it easier for the programmer to remember the purpose of the variable.

Example: The program in exercise 3, #3, could have been written:

```
10 M1 = 50
20 M2 = 65
30 M3 = 42
40 TOTAL = M1 + M2 + M3
50 AVERAGE = TOTAL / 3
60 PRINT TOTAL
70 PRINT AVERAGE
80 END
```

In what memory locations are the variables ~~TOTAL~~ and AVERAGE actually stored?

You must exercise caution with long variable names. This programmer tried to work out the total of three values:

```
10 NUM1 = 50
20 NUM2 = 65
30 NUM3 = 42
40 TOTAL = NUM1 + NUM2 + NUM3
50 PRINT TOTAL
60 END
```

Note: The name ~~TOTAL~~ was used instead of TOTAL. The latter would be illegal. The computer would consider it to be "TO", which is part of a BASIC command.

His answer turned out to be 126! Can you explain why?

### Storage Locations For Literal (String) Data

1. The computer has another set of storage locations with addresses such as:

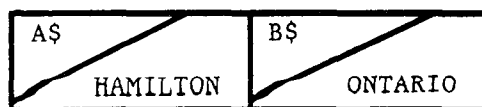
A\$, A0\$, A1\$, . . . , A9\$, AA\$, AB\$, etc.

These are the locations in which it stores its codes representing letters, words, etc.

2. Character strings can be stored by using the assignment statement. Note, however, that the string must be enclosed in double quotes (in the same way as messages were contained in PRINT statements).

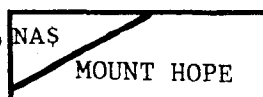
Example:    10 LET A\$ = "HAMILTON"  
             20 LET B\$ = "ONTARIO"  
             - OR -  
             10 A\$ = "HAMILTON"  
             20 B\$ = "ONTARIO"

The word LET is optional.



3. A string variable name can also be any length, but only the first two characters are actually used.

Example:    30 NAME\$ = "MOUNT HOPE"



actual address used

---

REVIEW QUESTIONS

1. What command should be typed in to clear the PET's memory of any existing programme?
2. Why do we number our programme lines in multiples of 10?
3. What is the largest allowable line number you can use on the PET?
4. Up to how many characters can a PET BASIC variable name contain?
5. With what type of character must a variable name begin?
6. If the storage location PETER contains the value 7, what value would the storage location PET contain?
7. Why can PALETTE not be used as a variable name?
8. What command requests the PET to print out all the statements that are stored in memory?
9. Make up simple programmes to solve the following problems. Type in your programmes and run them to see if they work.
  - (a) Store the value 7 in variable A1, and 11 in A2. Print out the value of the expression  $3(A1 + A2)^2$ .
  - (b) Store the value 77 in variable F and print out the value of the expression  $\frac{5}{9}(F-32)$ .
  - (c) Store the values of 7 in AB, 9 in AB1, and 10 in AB2 (in that order). Print out the value of  $AB + AB1 + AB2$ . Explain your result.





BRANCHING



" STATIC ELECTRICITY IS HARMFUL  
TO DISKS AND CIRCUITS. GROUND  
YOURSELF BEFORE USING COMPUTER  
DEVICES . "



TOPIC #5: BRANCHING

NEW

1. Turn on the PET and clear the screen. Type in the word NEW and press the RETURN key. When you turn off the PET, the memory is cleared, so that typing NEW is really not necessary. However, it is a good habit to develop for those occasions when you are working on the PET for an extended period of time and may be entering and running many different programmes without wishing to be constantly turning the PET off and on.

SQR

2. Type in the following programme:  
10 REM THE SQUARE ROOT FUNCTION  
20 X=2  
30 Y=SQR(X)  
40 PRINT "THE SQUARE ROOT OF";X;"IS";Y  
50 END

ARGUMENT

3. Now type in RUN to execute the programme. The programme assigns the value 2 to the variable X. Next the built-in function SQR( ) calculates the square root of the expression it finds within the brackets (This expression is called the ARGUMENT of the function), and assigns it to the variable Y. Line 40 provides the OUTPUT. When you examine the OUTPUT you should notice that the expression appearing in double quotes in line 40 is printed out per se but the values of X and Y are printed since they were not within the double quotes. Recall that a semicolon is used to separate expressions in a PRINT statement.
4. We wish to alter the above programme to print out the square root of not just 2, but of the subset of natural numbers 2, 3, 4, ... . To do this we could change lines 20 to X=3 and run the programme again. Then use X=4 and run it once more. This could be done over and over again but is very time consuming and hence not very efficient. An alternate method would be to have the programme itself increase the value of X and then have it go back and re-evaluate the square root and print it out.

INCREMENT

5. When you increase the value of a variable within a programme it is called INCREMENTING THE VARIABLE. To do this in the above programme, just type in the line:

45 X = X + 1

and press the RETURN key. Now use LIST to list the new programme. Notice that the line 45 has been automatically placed between lines 40 and 50. Line 45 takes the existing value stored in X, adds 1 (the increment) to it, and replaces it with this new value.

- GOTO           6. To get the programme to go back and calculate the square root of X we need a way of telling it to execute line 30 next. The GOTO statement allows us to do this. Type in the line:
- 47 GOTO 30
- and press the RETURN key. Now list the new programme. Again notice that line 47 has been inserted between lines 45 and 50. Before running this programme consider what it will do when executed: X will be assigned the value 2, Y will be assigned the value of the square root of 2 and both values will be printed. Then X will be increased by 1 to 3, its square root will be calculated and then both values will again be printed. This LOOP will be repeated over and over so that all the numbers 2, 3, 4, ... and their square roots will be printed.
- INFINITE LOOP 7. This is an INFINITE LOOP. The PET will continue to calculate and print these values until the values are too large for its memory or until it is instructed to stop.
- RUN/STOP       8. Whenever the PET goes into an infinite loop, you can stop the process by pressing the RUN/STOP key.
- BREAK           9. Type in the command RUN and press the RETURN key. Let the programme run for a few seconds then press the RUN/STOP key. You should get a BREAK message. The PET tells you the line it was executing at the moment that you pressed the RUN/STOP key.
- CONT.           10. To have the programme continue to run, type in the command CONT and press the RETURN key. The programme should continue where it left off, and again go into an infinite loop. Let it run for a few seconds and again stop it using the RUN/STOP key. Practice using the RUN/STOP key and the CONT command until you are familiar with them.
- LIST           11. By now, your programme has probably been scrolled right off the screen. Clear the screen and list your programme using the LIST command. Again, the whole programme should be listed on the screen. To list a single line on the screen, type LIST followed by the line number. For example, type
- LIST 30
- When you press the RETURN key, line 30 should be listed. To list a group of lines, type
- LIST 20 - 40
- When you press the RETURN key, lines 20, 30, and 40 should be listed on the screen. To list all the lines up to a given line, type
- LIST - 30
- When you press the RETURN key, lines 10, 20, and 30 should be listed. Similarly to list all the lines from a given line to the end of a programme, type
- LIST 30 -
- Now, when you press the RETURN key, lines 30, 40, 45, 47, and 50 should be listed.

---

### REVIEW QUESTIONS

1. What word is used for the expression found in the brackets following a built-in function?
2. Alter the programme above to increment X by 2 instead of 1, then run it for a few seconds.
3. Change line 20 to read 20 X = 10 and line 45 to read 45 X = X - 1. Now run the programme. Why did it stop this time?
4. Practice using the LIST command by getting the following statements listed:
  - a) line 45
  - b) lines 30 to 47
  - c) all the lines up to line 40
  - d) all the lines from line 40 to the end of the programme.
5. Type NEW and press the RETURN key. Now type LIST and press the RETURN key. What statements are listed and why?
6. Make up a programme to assign the value 1 to a variable, assign its square ( $1^2 = 1$ ) to a second variable, print the two values out, increase the first variable by 1 and go into an infinite loop whereby both the first variable and its square are printed. Run the programme for about twenty loops, and stop it. Now allow it to run for about 10 more loops.
7. Make up a programme to print the phrase HI THERE! Add one statement to your programme that will produce an infinite loop whereby the phrase HI THERE! will be printed over and over. Run the programme until a single column of HI THERE!'s appears on the screen.
8. Edit the programme in (7) above by adding a semicolon to the end of the PRINT statement. Run the programme again for a few seconds and notice what happens.





DECISION MAKING



"DON'T BEND YOUR DISKETTE.  
CREASES WILL DESTROY YOUR  
PROGRAM."



TOPIC #6: DECISION MAKING

1. Turn on the PET and clear the screen. Type in the following programme:

```
10 REM TABULATION OF A FUNCTION
20 X = 1
30 Y = X ↑ 2/(X-5)
40 PRINT "WHEN X IS";X;"Y HAS THE VALUE";Y
50 X = X+1
60 GOTO 30
70 END
```

2. This programme is designed to print out a table of values for the function defined in line 30. X will take the values 1, 2, 3, ... and for each X, Y will be calculated. There are two things wrong with this programme: One is that we once again have an infinite loop so we must be prepared to use the STOP key when we have printed enough values. The second problem is that when X takes the value of 5, the function defined in line 30 requires the PET to divide by zero. This will result in an error message. Run the programme.

IF...  
THEN...

3. To prevent division by zero we must avoid executing lines 30 and 40 when X takes the value 5. To do this type in the following lines:

```
25 IF X = 5 THEN GO TO 50
60 GOTO 25
```

If the expression  $X = 5$  is true in line 25 then the statement following THEN will be executed. If  $X = 5$  is false then control will pass to the line having the next higher number to 25, namely line 30. Retyping a line number results in the first version being replaced. When you list the programme using LIST you will see that the old line 60 has been replaced and that line 25 has been inserted between lines 20 and 30. Try running the programme again and use the RUN/STOP key to stop execution.

LOGICAL  
EXPRESSIONS

4. In the IF...THEN... statement, the expression between the IF and the THEN can be any logical expression. The statement following THEN can be any executable statement. (Note: When GOTO or GOSUB follow, the word THEN may be omitted.) Here are some examples:

```
10 IF A > 13 THEN B = A+5
20 IF B >= 2 THEN PRINT "WOW!"
30 IF C <= 5 THEN GOTO 100 (or: 30 IF C<=5 GOTO 100)
40 IF BOY <> GIRL GOTO 50
50 IF A < B AND B < C THEN? "TRUE"
60 IF X = 5 OR Y = 7 GOTO 100
```

Notice in 50 and 60 that expressions can be combined using the logical connectives AND and OR. In 40 the symbol for 'not equal to' is  $\neq$ . Line 20 will allow the expression "WOW!" to be printed only if B is greater than or equal to 2.

NOT

5. The logical word NOT may be used to negate a logical expression. For example:

```
70 IF NOT X > Y THEN GOTO 100
```

Now the statement following THEN will only be executed if  $X > Y$  is false. Thus 70 above is equivalent to:

```
70 IF X <= Y THEN GOTO 100
```

ENDING

6. We can use an IF...THEN... statement to provide a way of ending a loop after a finite number of steps. Replace line 60 in the programme of section (3) by typing:

```
60 IF X <= 10 THEN GOTO 25
```

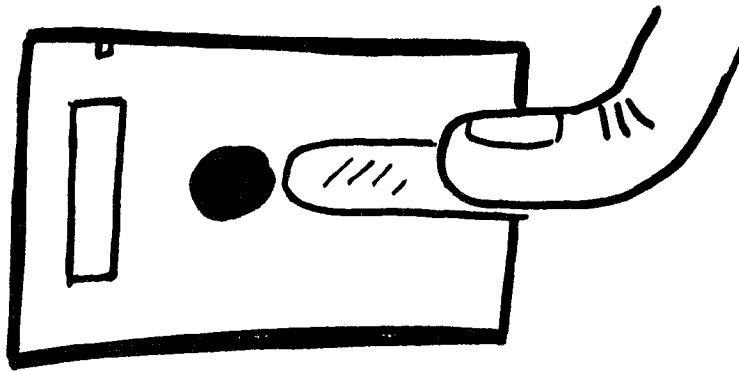
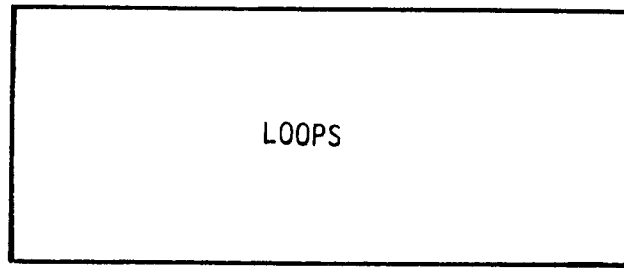
Now as long as X is less than or equal to 10, the programme will loop back to statement 25. As soon as X takes the value 11 control will pass to statement 70 which will end the loop. Now the programme should evaluate Y for  $X = 1, 2, \dots, 10$  except of course for  $X = 5$  for which Y is undefined. List the new version of the programme and run it.

---

#### REVIEW QUESTIONS

1. Translate each of the following statements into a BASIC IF....THEN...statement.
  - a) when X is less than 7 print TRUE
  - b) when SUM is greater than or equal to 11 print FALSE
  - c) if A is equal to B assign 3 to C
  - d) when X is less than 2 and at the same time Y is greater than 3 print STRANGE!
2. Use the logical NOT to write your answers to question (1) another way.
3. For each of the following problems, write the three BASIC statements that are necessary.
  - a) have X go from 1 to 10 in steps of 1 (that is the INCREMENT is 1)
  - b) have A go from -10 to 10 in steps of 2
  - c) have S go from 100 to 0 in steps of -1
  - d) have COUNTER go from 7 to 13 in steps of 1.
4. Write a BASIC programme to evaluate the expression  $D = B^2 - 4AC$  and to print it out. If D turns out to be negative, the PET should print 'IMAGINARY ROOTS'. If D is zero, the PET should print 'EQUAL ROOTS'. If D is positive, the PET should print 'TWO REAL EQUAL ROOTS'. Run your programme for each of the following sets of values and record the output.
  - a)  $A = 5, B = 7, C = -3$
  - b)  $A = 7, B = 3, C = 5$
  - c)  $A = 4, B = 4, C = 1$





"FINGERPRINTS DESTROY DATA.  
HANDLE DISKETTES CAREFULLY."



## TOPIC #7: LOOPS

1. Turn on the PET and clear the screen. Type in the following programme:

```
10 REM A SIMPLE LOOP
20 X = 1
30 PRINT X
40 X = X+1
50 IF X <= 10 GO TO 30
60 END
```

This programme prints out the numbers from 1 to 10. Run it.

- FOR...NEXT... STEP
2. In BASIC it is possible to replace lines 20, 40, and 50 with special looping statements. Type in the following lines and then list the programme now stored in memory.

```
20 FOR X = 1 TO 10 STEP 1
40 NEXT X
50
```

Notice that typing a line number without a following statement will cause the original statement to be erased from memory.

Statement 20 assigns 1 to X then causes the PET to execute all the steps between it and line 40. Line 40 causes X to be increased by the increment following the word STEP in line 20 (In this case, 1). Control passes back to line 20 and the loop is repeated until X assumes a value greater than that following the word TO in line 20 (In this case, 10). When the looping is complete, control passes to the next line number after the statement containing NEXT (In this case, line 60). Run the programme, and compare the results to those of the original one.

3. If the increment is 1, then the STEP 1 part of the FOR...NEXT statement may be omitted. Also, the variable name X may be omitted in the NEXT statement. Make these two changes, list the altered programme, and run it.

### **INDENTING**

4. In order to make a loop stand out more in a programme listing, the instructions within a loop should be indented. This is an accepted standard of good programming style. To indent in Commodore BASIC it is necessary to type a colon (:) after the line number. Then press the space bar to insert the number of blanks you desire, and follow with your instruction. Line up the beginning of all such instructions within the loop.

List the programme entered in (1) and (2), and insert the colon and extra blanks in line 30.

```
10 REM A SIMPLE LOOP
20 FOR X=1 TO 10 STEP 1
30: PRINT X
40 NEXT X
60 END
```

#### MULTIPLE STATEMENTS

5. By using a colon to separate statements, it is possible to combine many statements on a single line. A single line in BASIC contains up to 80 characters. Since the screen on the PET is only 40 characters wide, it is possible to continue typing a statement onto the next line of print. The RETURN key should be pressed only after all characters in the statement have been typed, even if the statement has overlapped onto the next line. You may not, however, extend a statement beyond two lines on the PET. Type NEW and then key in the following programme:

```
10 REM A SIMPLE LOOP
20 FOR X = 1 TO 10:PRINTX:NEXT
30 END
```

Run the programme. You will notice that we have combined the old lines 20, 30, and 40 into the new line 20. Line 20 now contains a complete loop, each statement of which is separated by colons.

6. So far the output has been vertical. To get the numbers from 1 to 10 printed horizontally, use the INST/DEL key to insert a semicolon between the X and the colon:

```
20 FOR X = 1 TO 10:PRINTX;:NEXT
```

The semicolon will prevent the automatic carriage return from operating, so that all the output will appear on a single line.

7. Type NEW and then key in the following programme:


```
10 REM LOVER
20 FOR I = 1 TO 400:PRINT"LOVE";:NEXT I
30 END
```

Run the programme. The screen should gradually fill with LOVE. Since we are attempting to print LOVE 400 times and since the screen can only hold 1000 characters (25 X 40 = 1000), the screen cannot hold all the output at one time. Thus when the screen is filled, the top line scrolls off the screen to make room for the new bottom line. This scrolling will continue until all output has been printed.

## REVIEW QUESTIONS

1. Alter the programme in part 7 to provide a single vertical column of LOVE.
2.
  - a) Alter the example on page 10 to print out the square roots from 2 to 20 (using FOR-NEXT).
  - b) Modify it to print the roots from 10 to -10 (in that order). Run it to see what happens.
3. Recall that it is possible to write 1 line programmes for the PET (Do not use line numbers and use the RETURN key to execute the programme instead of typing RUN). Write a one line programme that would cause the PET to print your name 200 times. First have your name appear in a vertical column, then rewrite the programme to have the screen fill with your name written over and over.
4. Rewrite this program to perform the looping using FOR-NEXT statements. Indent the instructions inside the loop. Enter it on the computer, LIST it, and RUN it.

```
10 REM.... A TABLE OF VALUES
20 PRINT " Y = X↑2-5*X"
30 PRINT: PRINT "X-VAL Y-VAL"
40 X = -5
50 Y = X↑2-5*X
60 PRINT X;Y
70 X = X+1
80 IF X <= 5 THEN GOTO 50
90 END
```

Recall the meaning  
of  from page 5



EXERCISE

- (5) How many times will each of these loops be repeated? Try them on the PET to see if you are right.

(a) 10 FOR I = 10 TO 50 STEP 5  
20 PRINT I  
30 NEXT I

(b) 10 FOR K = 3 TO 14 STEP 2  
20 PRINT K  
30 NEXT K

(c) 5 LET N = -1  
10 FOR M = 10 TO 1 STEP N  
20 PRINT M  
30 NEXT M

- (6) Try this example on the screen:

```
10 FOR J = 10 TO 0 STEP -1
20 PRINT J
30 NEXT J
40 PRINT "BLAST OFF!!"
50 PRINT
60 END
```

The computer will do this so fast that it all seems to appear almost simultaneously! To make the count down go more slowly, we have to somehow slow the computer down. This brings to mind another use for loops - a "time delay loop".

Example: 25 FOR K = 1 TO 500  
26 NEXT K

Nothing is being done in this loop! We are simply killing time by making the computer count to 500!

Add these lines to the previous program and run it again. Observe the new result.

- (7) Try this example on the PET:

```
10 PRINT "▼"
20 X = INT(RND(0)*50)
30 Y = INT(RND(0)*50)
40 PRINT "WHAT IS"; X; "TIMES"; Y; "?"
50 INPUT Z
60 IF Z = X*Y THEN PRINT "CORRECT!"
70 IF Z <> X*Y THEN PRINT "WRONG!"
80 GOTO 10
```

There is a problem with all such programs: After the message is printed in lines 60 or 70, control goes back up to 10 and the screen is cleared.

This happens so quickly that the message can't even be read!

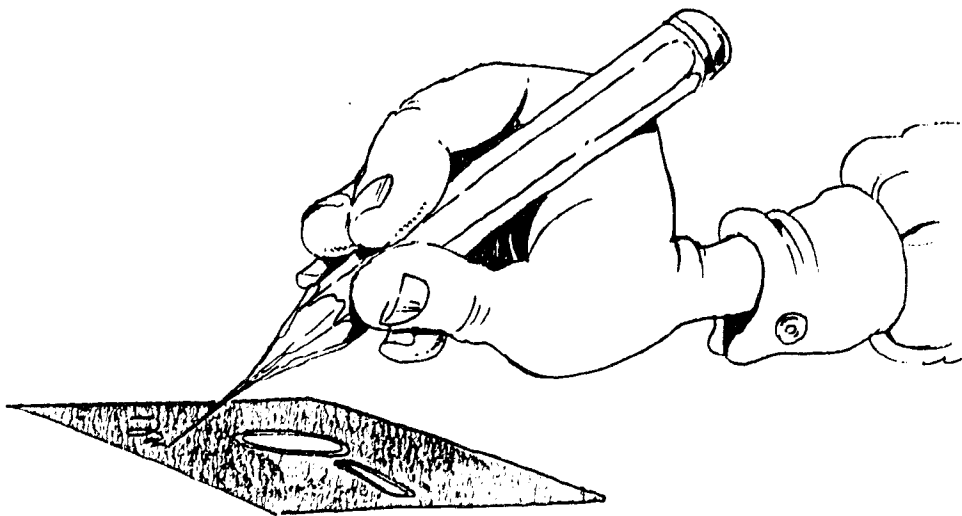
SOLUTION: (a) Use either a "screen holding " technique

- OR -

(b) Use a time delay loop.

Experiment in this case with a time delay loop, using different final (test) values. What value will make a delay for about five seconds? Which method (a or b) do you think is more appropriate for programs designed for students?

## INTERACTIVE BASIC



"USE FELT PENS FOR LABELS. BALLPOINT PENS AND PENCILS CAN RUIN A DISK BY PRESSING HARD ON THE DISK SURFACE."



TOPIC #8: INTERACTIVE BASIC

INPUT

1. In all the programmes we have run to this point, variables have been assigned values using assignment statements. Whenever we wanted to assign different values we had to rewrite the assignment statements. The INPUT statement allows us to request information from the keyboard and thus to assign values to variables externally.

2. Type in the following programme:

```
10 REM*USING INPUT STATEMENTS*
20 INPUT A:INPUT B
30 C = A+B
40 PRINT A;"+";B;"=";C
50 END
```

When you run this programme a ? will appear on the screen. This ? is prompting you to enter a number on the keyboard. When you do enter such a number, say 7, and then press the RETURN key, this value 7 will be stored in A. A second ? will now appear, prompting another entry from the keyboard. Enter a 3, then press the RETURN key. 3 will be stored in B and the remainder of the programme will be executed. Run the programme.

3. We can make the prompting a little more meaningful by using extra PRINT statements. Type in the following lines:

```
15 ? "TYPE THE VALUE OF A";
20 INPUT A
22 ? "TYPE THE VALUE OF B";
25 INPUT B
```

List the new programme and run it. The semicolons force the ?'s to appear at the end of each line of output giving a more meaningful display.

4. The same thing can be accomplished using just INPUT statements. Type in the following lines:

```
15
20 INPUT "TYPE IN A";A
22
25 INPUT "TYPE IN B";B
```

Again notice that typing a line number with no following statement erases the original statement. List the new programme and run it.

?:?:?

5. When programmes requesting information are run, it is nice to clear the screen before displaying the request. Type in the following programme (Remember to type in NEW first):

```
10 REM* SIMPLE TEST*
20 PRINT "❤ PICK A NUMBER FROM 1 TO 10";
30 INPUT A
40 ??:?
50 PRINT "PICK ANOTHER NUMBER FROM 1 TO 10";
60 INPUT B
70 ??:?
80 PRINT "WHAT IS ";A;"+";B;
90 INPUT C
100 IF C = A+B THEN ? "SUPER!":GOTO 120
110 IF C <> A+B THEN ? "SORRY THE ANSWER IS";A+B
120 END
```

In line 20 the ❤ is the symbol for CLEAR SCREEN (CLR/HOME and SHIFT). In lines 40 and 70, three ?'s or PRINTS are used to provide vertical spacing. ??:? causes three blank lines to be displayed. Notice in 100 and 110 the logical expression can contain arithmetic operations. Is the :GOTO 120 in line 100 really necessary in this programme? Run the programme a few times. Notice that the screen clears during each run.

TIME DELAY

6. To have this programme run itself, say 5 times, we could use a loop. Type in the following lines:

```
15 FOR I = 1 TO 5
120 NEXT I
130 END
```

List the new programme and run it. You should notice that after each question has been answered, you do not get a chance to read the result before the screen clears. This can be prevented by using a time delay loop. Type in the following lines:

```
120 FOR J = 1 TO 2000:NEXT J
125 NEXT I
```

Line 120 is a loop which causes the variable J to take values from 1 to 2000. This is all that happens within the loop, but 2000 steps require the PET to work for a few seconds giving you a chance to read what is on the screen before statement 20 is executed and the screen is cleared. List the new programme and run it.

7. Change the 2000 in line 120 to 1000 and run the programme. Try various other values for this number until you find one which gives you enough time to read the display before the screen is cleared.

---

### REVIEW QUESTIONS

1. What type of statement allows us to assign values to variables from the keyboard while a programme is running?
2. How can you get the ? to appear at the end of the prompting statement, instead of on the next line, when using INPUT statements?
3. What statements would cause the PET to output 5 blank lines?
4. Write a BASIC line that would create a time delay of approximately 3 seconds.
5. Write a BASIC programme that requests the user to select two numbers from 1 to 10 and then requests their product. Your programme should evaluate this answer and give an appropriate response.
6. Change the programme in question 5 to request 10 different sets of values and responses.
7. Add lines 35 and 65 to the final programme of this topic to test the value that has been input via the keyboard. If the value is not in the range from 1 to 10, these statements should pass control back to statements 20 or 50 respectively requesting values that are in this range.



EXERCISE

- (8) (a) Enter this program on the PET:

```
10 PRINT "SIMPLE INTEREST CALCULATOR"
20 PRINT
30 PRINT "ENTER VALUE FOR PRINCIPLE"
40 INPUT P
50 LET R = .05
60 LET T = 3
70 I = P*R*T
80 PRINT "INTEREST"
90 PRINT I
100 END
```

Run it a few times, with different responses, to determine the interest after 3 years at 5% per annum.

- (b) Modify the program so that you are asked to enter the rate (as a decimal) and the time (in years) instead of having them assigned in the program.

- (9) Write a program to convert a Celcius temperature input by the user into its equivalent Fahrenheit temperature.

$$F = \frac{9}{5} C + 32$$

- (10) Consider this program:

```
10 PRINT "PRACTICE DRILL FOR ADDITION"
20 PRINT "INPUT TWO NUMBERS"
30 INPUT A, B
40 PRINT "WHAT IS THEIR SUM?"
50 S = A + B
60 INPUT C
70 IF C = S THEN 100
80 PRINT "WRONG - TRY AGAIN"
90 GOTO 40
100 PRINT "CORRECT"
110 END
```

Modify this program at line 70 so that it asks:

"if C is not equal to S then . . ."

Make appropriate related changes to lines 80-100.

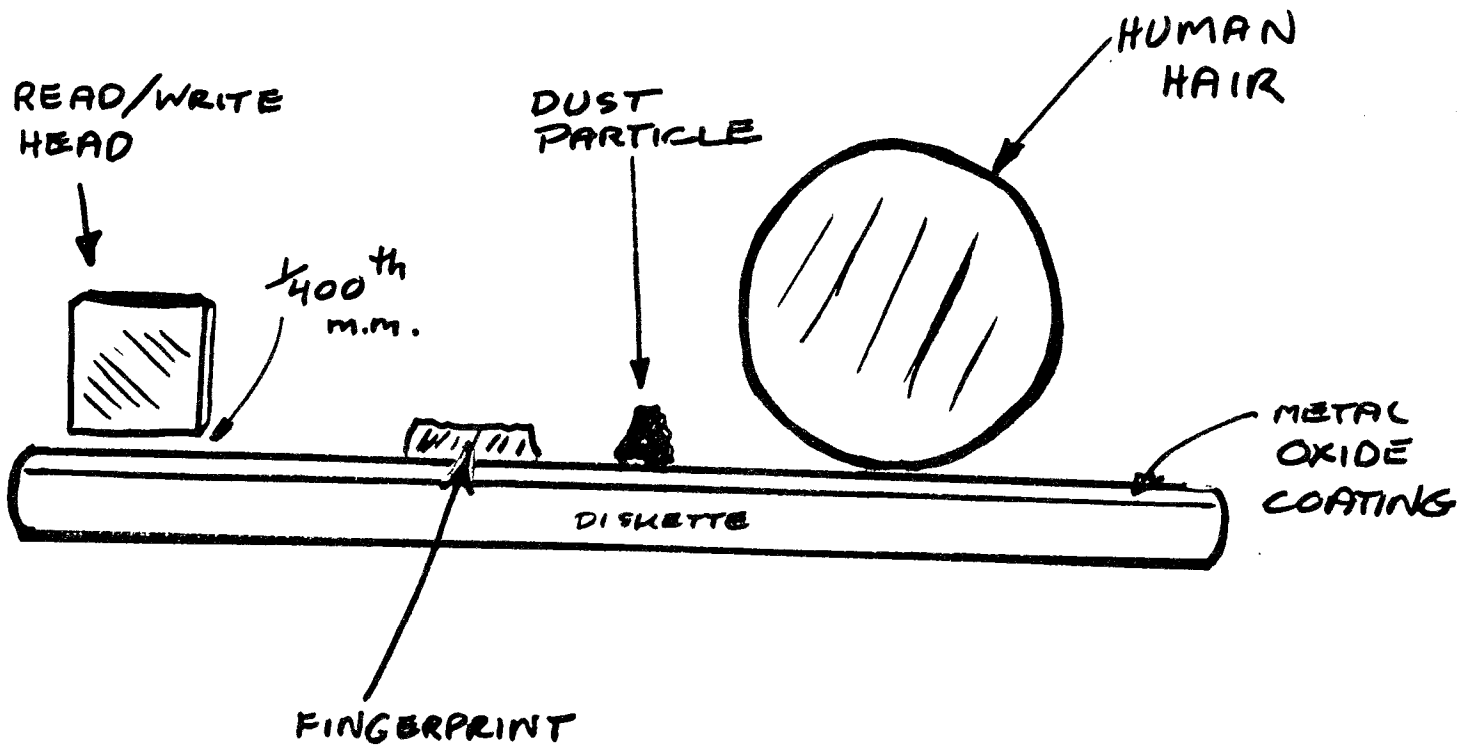
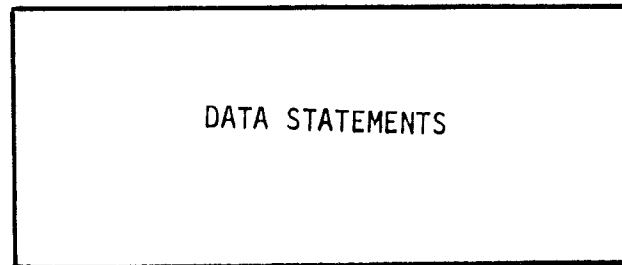
- (11) Write a guessing game program in which the user will attempt to guess a number from 1 to 100 (the computer's secret number will have to be assigned in the program). His guesses will receive clues such as:

TOO LOW - TRY AGAIN      or      TOO HIGH - TRY AGAIN

If guessed correctly, print

RIGHT ON!

and then end the program.



"NOTE THE EFFECT OF CONTAMINANTS  
ON A DISKETTE."



TOPIC #9: DATA STATEMENTS

READ...DATA  
RESTORE

1. We have seen two ways of assigning values to variables: Assignment statements and INPUT statements. A third method is to read information from a DATA statement. To do this we use both the READ and DATA statements. Type in the following programme:

```
10 REM *ANOTHER SIMPLE TEST*
20 FOR I = 1 TO 5
30 : READ A,B
40 : PRINT "☐"; A;" + ";B;"=";
50 : INPUT C
60 : IF C = A+B THEN PRINT "CORRECT"
70 : IF C<>A+B THEN PRINT "SORRY, THE ANSWER IS"; A+B
80 : FOR J=1 to 2000: NEXT J
90 NEXT I
100 DATA 3, 7, 4, 4, 5, 1, 6, 9,8, 2
110 END
```

The loop from statement 20 to 90 is executed 5 times. During each execution A and B take values from the DATA statement: First A is 3 and B is 7, then A is 4 and B is 4, etc. Line 40 clears the screen and requests the sum of A and B. When the answer is typed in, it is stored in C and the PET evaluates it. Statement 80 is a time delay loop. After 5 runs the loop ends. At this point the data has all been used up.

2. Multiple DATA statements may be used (Especially when the amount of data would require more than 80 characters). Replace statement 100 with:

```
100 DATA 3, 7, 4, 4, 5
105 DATA 1, 6, 9, 8, 2
```

Run the programme and compare the results with the original. Keep this programme in memory for item (5).

STRING  
VARIABLES

3. String variables can be used to store string constants. A string constant is any combination of up to 255 characters. These characters can be graphic, numeric, alphabetic, and even some of the special characters available. String variable names follow all the rules covered for real variable names (i.e. variables used to store real numbers) in topic #3, with the one exception that they must be followed by a \$ sign. String constants must be enclosed by double quotation marks.

```
examples: A$ = "FUN!"
 BOY$ = " ♥♥♥♥♥ "
 C$ = "5"
 D$ = "6"
 ANSWER1$ = "THAT IS CORRECT."
```

CONCATENATION 4. When string constants are 'added' it is called CONCATENATION (see topic #2). Thus, if C\$ and D\$ were assigned values as in the examples above,

SUM2\$ = C\$ + D\$

would have the value "56" not 11.

5. Add these lines to the program:

```
92 ? "WOULD YOU LIKE TO TRY AGAIN?"
93 ? "ANSWER YES OR NO, PLEASE."
94 INPUT AN$
95 IF AN$ = "YES" THEN GOTO 20
96 IF AN$ = "NO" THEN GOTO 110
97 GOTO 93
```

Run it, and answer YES to the request to try again. What happens? Since the data was used up by the first run through the program, there was none left to be read the next time through.

RESTORE 6. Edit line 95 to the following:

```
95 IF AN$ = "YES" THEN RESTORE: GOTO 20
```

Run it again, and answer YES once more. What happens this time? The RESTORE command sets up the original data so that subsequent READ statements will start reading the original data from the beginning. The RESTORE command is necessary for examples like this where an exercise may be repeated by the user.

7. Type in the following programme:

```
10 REM *TEST #3*
20 FOR I = 1 TO 5
30 : READ A$,B$
40 : ?"WHAT IS THE CAPITAL OF ";A$;
50 : INPUT C$
60 : ?:::
70 : IF B$=C$ THEN ?"CORRECT!"
80 : IF B$<>C$ THEN ?"SORRY, THE ANSWER IS ";B$
90 : FOR J=1 TO 2000: NEXT J
100 NEXT I
110 DATA "CANADA","OTTAWA","ENGLAND","LONDON",
 "THE UNITED STATES","WASHINGTON"
120 DATA "ONTARIO","TORONTO","FRANCE","PARIS"
130 END
```

Notice we read the country and the correct answer right from the DATA statement. Then, when a response is entered, we can evaluate it. Run the programme.

## REVIEW QUESTIONS

1. Change lines 110 and 120 in the last programme of this topic, omitting the double quotes. Run the new programme. Does it still work?
2. What statement is used to reset the data bank at its original position?
3. If B\$ = "1" and O\$ = "0" what would BO\$ = B\$ + O\$ be equal to? What is the name given to this operation?
4. Change the programme in section 5 of this topic to ask the question WHAT IS THE PAST TENSE OF 'VERB'? where 'VERB' is a verb to be read from a DATA statement (the DATA statement should also supply the correct answer). Again this programme should ask five different questions, and evaluate the answers given.
5. Change the programme in section 5 of this topic to ask 5 similar questions from each of the following areas:
  - 1) French (For example: It might ask "What is the french word for WORD?", where WORD is one of the english words stored in the DATA statement).
  - 2) Chemistry (For example: It might ask "What is the chemical symbol for ELEMENT?", where ELEMENT is one of the constants stored in the DATA statement).
  - and 3) Astrology (For example: It might ask "What is the symbol for SIGN?", where SIGN is one of the astrological signs stored in the DATA statement).
6. By storing the entire question in the data statement, it is possible to ask completely different questions in the same programme. Write the programme which would make use of the following data statements to ask questions and evaluate the responses:  
  
110 DATA "7+13 = ", "20", "HOW MANY SYLLABLES DOES  
      'ROCKET' HAVE", "2"  
  
120 DATA "WHAT IS THE CUBE ROOT OF 27", "3", "WHAT  
      IS THE FORMULA FOR VOLTAGE DROP", "V = IR"  
  
130 DATA "WRITE APRIL 15, 1948, IN METRIC FORMAT",  
      "1948 04 15"  
  
7. Could the double quotes be omitted from the DATA statement in line 130 above? Explain what would happen if they were omitted.





SOME BUILT-IN FUNCTIONS



"Tell me more about 'bytes'!"



TOPIC #10: SOME BUILT-IN FUNCTIONS

RND            1. A fourth method of assigning values to variables (See topic #7 for  
the first three) is by use of the random number generator.

INT            Type in the following programme:

```
10 REM*GENERATING RANDOM NUMBERS*
20 FOR I = 1 TO 5
30 : A=INT (RND(0)*9+1)
40 : B=IN (RND(0)*9+1)
50 : ?" ";A;"-";B;"=";
60 : INPUT C
70 : IF C=A-B THEN ?"O.K."
80 : IF C<>A-B THEN ?"WRONG! THE ANSWER IS";A-B
90 : FOR J = 1 to 2000: NEXT J
100 NEXT I
110 END
```

Run the programme.

2. The built in function RND ( ) generates a random number between 0 and 1. The argument, or expression in brackets, determines the type of number selected:

- 1) If the argument is negative, the first number in a new set of random numbers is selected (e.g. RND(-2)).
- 2) If the argument is positive, the next random number in the existing set of random numbers will be selected.
- 3) If the argument is zero, a random number based on the value of the built in time clock will be produced.
- 4) To achieve almost 'perfect' randomness, you can combine 1) and 3) (e.g. RND(-RND(0))) randomly selects a number from a randomly selected sequence of random numbers).

Since the random number selected is between 0 and 1, multiplying by 9 will select a number greater than 0 and less than 9. Adding 1 will produce a decimal value greater than 1 and less than 10. The INT ( ) function truncates (drops) the decimal part of any value. Thus

INT(RND(0)\*9+1)

should produce a number in the set 1, 2, 3, ..., 9.

3. Change lines 30 and 40 to read:

```
30 A = INT(RND(0)*20+5)
```

```
40 B = INT(RND(0)*20+5)
```

List the new programme and run it. What set of values are generated now?

4. Change lines 30 and 40 to read:

```
30 A = INT(RND(0)*11-5)
```

```
40 B = INT(RND(0)*11-5)
```

List the new programme and run it. What set of values are generated now?

5. To generate values in the range N to M use the following format:

```
INT(RND(0)*(M-N+1)+N)
```

According to this formula, A and B should be elements of the following sets:

in section (1):  $1 \leq A \leq 9$ ,  $1 \leq B \leq 9$  ( $9-1+1=9$ )

(3):  $5 \leq A \leq 24$ ,  $5 \leq B \leq 24$  ( $24-5+1=20$ )

(4):  $-5 \leq A \leq 5$ ,  $-5 \leq B \leq 5$  ( $5-(-5)+1=11$ )

INTEGER  
VARIABLES

6. We have seen numeric variables (topic 3) and string variables (topic 8). The third and last type is integer variables. These are used to store numbers which will always be integers (no decimal part). The variable name must be followed by a % sign.

```
10 A=23
```

```
20 B=6
```

```
30 C%=A/B
```

In this example, C% will store the value 3. It can only be assigned the integer part of the answer 3.83333333 (the decimal part is chopped off, or ignored).

Thus, integer variables can be used in place of the INT ( ) function. The example in part (1) would work the same if variables A, B, and C were replaced by A%, B%, and C% - and lines 30 and 40 were changed to:

```
30: A%=RND(0)*9+1
```

```
40: B%=RND(0)*9+1
```

GET

7. In the examples so far, a time delay loop was used to give us time to read the display on the screen. To give us complete control of when the screen will clear we can use the GET statement. Change the above programme by typing in the following lines:

```
85: ???
```

```
90: ?""PRESS ANY KEY TO CONTINUE."
```

```
95: GET A$: IF A$=""GOTO 95
```

Statement 90 gives us the information necessary to proceed with the programme. Line 95 causes the PET to go to the keyboard and accept the character of the next key pressed. This will be stored in the STRING variable A\$. The second statement in line 95 compares the character stored in A\$ with 'nothing' (see the double quotes together). If no key has been pressed yet, then nothing (i.e., no character) will be stored in A\$. Thus the expression A\$="" will be true and control will pass back to the GET statement. This again is an infinite loop, and the PET will stay in this loop until some key is pressed. When any key is pressed, command will pass to the next statement, namely 100, and the programme will continue.

Press only one key, as instructed. If you press two, the first key will be read by GET and stored in A\$. The second key will stay in the PET's input buffer (an area that collects typed in data) and will be used as the first key (or value) in your next answer (i.e., in line 60). If your second key is the RETURN key, this will stop your programme at line 60 (a RETURN, typed as an answer to an INPUT, will stop your programme).

Run the new programme. Experiment with it.

---

#### REVIEW QUESTIONS

1. Write the random functions necessary to generate numbers in the following ranges:

- a)  $1 \leq X \leq 10$
- b)  $1 \leq X \leq 25$
- c)  $1 \leq X \leq 6$  (for use when simulating the rolling of a die)
- d)  $1 \leq X \leq 52$  (for use when simulating the drawing of a card from a deck of 52)
- e)  $-10 \leq X \leq 10$
- f)  $-10 \leq X \leq -1$  (hint generate  $1 \leq X \leq 10$  and multiply by -1)

2. If lines 90 and 95 were changed (in the programme of this topic) to:

90: ? "ARE YOU READY TO CONTINUE?"

95: GET B\$:IF B\$ <> "Y" GOTO 95

the PET would continue executing the programme as soon as the Y key were pressed. If any other key were pressed, control would remain at line 95. Change lines 90 and 95 to handle the question: "DO YOU NEED MORE TIME?"

3. So far the 'TEST' programmes have generated just 5 questions during each run. Refer to the programme of this topic. Type in the following lines:

20

90 ? "DO YOU WANT TO TRY ANOTHER QUESTION?"

95 GET C\$:IF C\$="" GOTO 95

100 IF C\$="Y" GOTO 30

List and run the new programme. Notice that it generates as many questions as desired.

4. a) Type in the following programme and run it a few times:

```
10 REM * MAKING CHANGE *
20 ? " PLEASE ENTER AN AMOUNT LESS THAN $1"
30 ? "(ENTER THE CENTS WITHOUT DECIMAL POINT)"
40 ? "AMOUNT";
50 INPUT A%:IF A%>100 THEN GOTO 20
60 Q%=A%/25
70 R%=A%-25*Q%
80 D%=R%/10
90 R%=R%-10*D%
100 N%=R%/5
110 P%=R%-5*N%
120 ? :? A%;"CENTS CAN BE MADE UP OF"
130 ?Q%;"QUARTERS,";D%;"DIMES"
140 ?N%;"NICKELS,";P%;"PENNIES"
150 END
```

Statement 50 determines the number of quarters by dividing the original amount A% by 25. Because Q% is an integer, only the non-decimal part of the answer will be stored in Q%. Statement 60 determines the amount of money remaining after the quarters have been removed. This pattern repeats for the dimes (D%), nickels (N%), and pennies (P%).

- b) Write a programme similar to the one in (4a), but use regular numeric variables instead of Integer variables. You will have to utilize the built-in INT( ) function. The problem is to convert a number of dollars less than \$100 into \$50, \$20, \$10, \$5, \$2, and \$1 bills.

- (5) Type and run the following program:

```
10 REM PROGRAM TO TEST ADDITION SKILLS
20 REM TEN QUESTIONS
30 A = 1
31 REM "A" WILL COUNT NUMBER OF QUESTIONS DONE
40 N1 = INT (100*RND(0))
50 N2 = INT (100*RND(0))
60 PRINT "GIVEN THESE NUMBERS:"
61 PRINT N1
62 PRINT N2
63 PRINT "WHAT IS THEIR SUM?"
70 INPUT ANS
80 IF ANS = N1 + N2 THEN PRINT "RIGHT!"
90 IF ANS < > N1 + N2 THEN PRINT "WRONG!"
100 A = A + 1
110 IF A <= 10 THEN 40
120 END
```

- (6) How would you modify the previous program to:

- (a) Run through 50 questions?
- (b) Use integers from -50 to +50?
- (c) Do multiplication questions?
- (d) Stop the program when the student decides he has done enough questions?
- (e) Keep track of the number right and wrong?

- (7) Write a program to calculate the mass of an object given its density (random number between 2 and 7) and volume (random number between 10 and 30).

$$M = D \times V$$

At the beginning of the program, ask the student how many questions he would like to do. At the end, indicate the percentage correct.

- (8) If problem (2) had been given to calculate the density given a mass (50-150) and volume (10-30) using:

$$D = \frac{M}{V},$$

the user may not have gotten any questions correct, because his answer probably wouldn't equal the computer's (how many decimal places does the computer work it out to? Who knows?). In such questions, the user should be asked to give his answer rounded off to 2 decimal places, say.

Thus, the programmer will need a technique to round off answers generated by the computer:

```
Example: 100 D = M/V
 110 D = INT (100*D + .5)/100
```

Explanation: Say  $M = 60$  and  $V = 21$   
Then density = 2.85714...

If we did  $\text{INT}(D)$  we'd end up with 2. We want two decimal places, so first move the decimal point over two places.

$$100 * D$$

Now  $\text{INT}(100 * D)$  would be 285. If we had rounded, the chopped off 7 would have made our number 286. That is the purpose of the .5 -- to make any trailing digit of .5 or greater have an effect before the chop off.

i.e.:

$$\begin{aligned} 100 * D + .5 &= 100 * 2.85714 + .5 \\ &= 285.714 + .5 \\ &= 286.214 \end{aligned}$$

$$\begin{aligned} \text{Now } \text{INT}(100 * D + .5) / 100 \\ &= 286 / 100 \\ &= 2.86 \end{aligned}$$

The division by 100 moved the decimal back where it belonged.

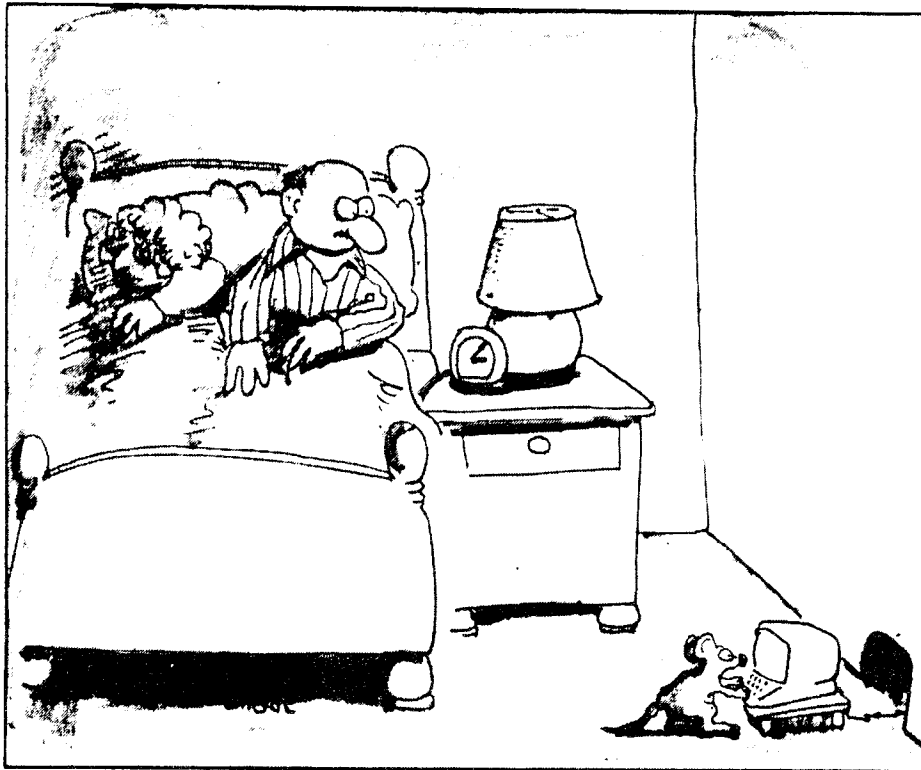
In general:

TO ROUND A VALUE OFF TO N DECIMAL PLACES, USE:

$$\text{INT} (10 \uparrow N * \text{NUMBER} + .5) / 10 \uparrow N$$



## SUBROUTINES





TOPIC #11: SUBROUTINES

- TAB
1. Type in and run this short programme:

```
10 ?"0123456789"
20 ? TAB(5);"X"
30 ? "X WAS PRINTED BY A TAB(5)"
40 END
```

The function TAB( ) sets the cursor to the print position determined by the argument - as long as you realize that the PET calls the first print position 0, the second 1, etc.
- GOSUB  
RETURN
2. In some programmes an identical series of instructions may have to be used many times. By using the BASIC statements GOSUB and RETURN, IT IS POSSIBLE TO WRITE THESE REPEATED INSTRUCTIONS ONLY ONCE. Type in the following programme:

```
10 REM*THE TITLE PROGRAMME*
20 ? "☐":??:??:?TAB(14);"THE TITLE"
30 ? TAB(14);"☐☐☐ ☐☐☐☐"
40 GOSUB 1000
50 ? "☐":??:??:?TAB(11);"WRITTEN BY: R AUTHOR☐"
60 GOSUB 1000
70 ? "☐":??:??:?"THAT'S ALL FOLKS!"
80 END
1000 FOR I = 1 TO 2000:NEXT I
1010 RETURN
```

In line 20 we use multiple ? statements to obtain vertical spacing. Statement 30 provides underlining for the title. Note the uses of the TAB function. The ☐ symbol is obtained by pressing the # key while also pressing the shift key. Statement 40 directs the PET to execute line 1000 next. Line 1000 is of course the time delay loop we have used before. Line 1010 then directs the PET to RETURN to the line immediately following the GOSUB statement. Thus control passes to line 50. In line 50 the R and ☐ symbols turn the reverse field function on and off. Line 60 is much like line 40 except that when control returns from the subroutine, statement 70 will be executed. Notice that an END statement should precede all subroutines. Run the programme.
  3. Change line 50 in the above programme by replacing the word AUTHOR by your own name. Adjust the argument of TAB to centre the new output on the screen.

4. For a more interesting effect, type in the following lines:

```
45 FOR J = 1 TO 5
55 FOR K = 1 TO 500:NEXT K
57 NEXT J
```

List and run the new programme. It is a good programming technique to print out a 'title page' for all interactive programmes. The above programme offers a typical example of such a title page.

TIME #

5. The PET has a built-in time clock which keeps very accurate time. In the PET microcomputer, time is measured in JIFFIES with 60 JIFFIES equal in length to 1 second. Each time the PET is turned on, the time clock begins running. To set the clock at the correct time, follow the following procedure: 1) type in

```
TIME$ = "HHMMSS"
where HH is the hour between 00 and 23
 MM is the minute between 00 and 59
 and SS is the second between 00 and 59
(Notice the double quotes - do not forget them!)
```

2) the 'second' value typed in above should be about 10 seconds later than the actual time. When the actual time is reached, press the RETURN key to activate the clock.

Set the clock on the PET at the correct time.

LEFT\$  
RIGHT\$  
MID\$

6. To find out the time from the PET, type in the following line:

```
? TIME$
```

The answer will again appear in the form HHMMSS. After setting the clock, type in the following programme, and run it:

```
10 ?"▼":REM*MY PET CLOCK*
20 TS = TIME$
30 HS = LEFT$(TS,2)
40 MS = MID$(TS,3,2)
50 SS = RIGHT$(TS,2)
60 PRINT " 5 a a a a a a ";HS;":";MS;":";SS
70 GOTO 20
```

Statement 30 contains the function LEFT\$( , ). This function takes the number of characters specified in its second argument from left side of the string specified in the first argument. Thus if TS = "172513", then LEFT\$(TS,2) would be "17". Statement 50 contains the function RIGHT\$( , ). This function takes the number of characters specified in its second argument from the right side of the string specified in the first argument. Thus if TS = "172513", then RIGHT\$(TS,2) = "13". Statement 40 contains the function MID\$( , , ). This function takes the number of characters specified in its third argument from the string specified in its first argument starting at the character named in its middle argument. Thus if TS = "172513", then MID\$(TS,3,2) = "25".

Line 60 contains the symbol **S** which is a cursor control symbol referring to the CLR/HOME key, and the symbol **A** which refers to the **↑/CRSR/↓** key. Thus the time is split up into hours, minutes, and seconds.

TIME

7. The number of seconds required for a response can be determined using the TIME variable. Type in the following programme and run it:

```
10 REM*HOW FAST DO YOU ADD?*
20 A = INT (RND(0)*9+1)
30 B = INT (RND(0)*9+1)
50 ? " S ";A;"+";B;"=";
60 T = TIME
70 INPUT C
80 S = INT ((TIME-T)/60)
90 IF C = A+B THEN?"RIGHT! IT TOOK YOU";S;"SECONDS."
100 IF C <> A+B THEN?" SORRY, THE ANSWER IS";A+B
110 ? "DO YOU WANT ANOTHER QUESTION?"
111 ? "PRESS Y OR N. DO NOT PRESS 'RETURN' ."
115 GET C$:IF C$ = ""GOTO 115
120 IF C$ = "Y" GOTO 20
130 END
```



The number of JIFFIES that have elapsed, since the PET was turned on, is stored in TIME. In statement 60 we record the value of TIME in T, at the instant the question appears on the screen. Statement 80 is executed at the instant that the RETURN key is pressed. Thus by subtracting the value of T (old time) from the current value of TIME, we can find the number of JIFFIES that have elapsed. Dividing by 60 converts the JIFFIES into seconds, and using the INT( ) function truncates the decimal part.

---

#### REVIEW QUESTIONS

1. On the programme of section #1 why could we not use GOTO 1000 in lines 40 and 60 instead of GOSUB 1000?
2. In the programme of section #1 we used multiple ? statements to get vertical spacing. Re-write lines 20, 50, and 70 using cursor controls to achieve the same result.
3. Describe the output you would expect from the following programme:

```
10 ? " S "
20 FOR I = 0 TO 39
30 PRINT " S A A A";TAB(I);"*"
40 FOR J = 1 TO 100:NEXT J
50 NEXT I
```

Run the programme. If the  in line 30 were changed to  , what would happen? Make the change, and run the new programme. Can you suggest a way to speed up or slow down the '\*'? Run the programme at half the original speed and then at twice the original speed (Hint: see line 40).

4. What is the difference between TIME\$ and TIME in PET BASIC?
5. Take the programme of section #5, and add the following lines:

```
15 GET A$: IF A$ = ""GOTO 15
70 GOTO 15
```

What effect should these lines have on the programme? List the new programme and run it. If nothing happens, press any key.

6. Write a programme similar to the one in (4a) of Topic #9, but have it convert the time stored in TIME (in JIFFIES) into hours, minutes, and seconds. Your output should appear as a series of statements such as:

```
EX: TIME = 954420 JIFFIES
 HOURS = 4
 MINUTES = 25
 SECONDS = 7
```



EXERCISE

- (7) Rewrite this program using a subroutine to eliminate unnecessary duplication of code in lines 80-150 and 200-270:

```
10 RIGHT = 0
20 PRINT "♥ METRIC CONVERSION"
30 PRINT
40 PRINT "(1) 150 CM EQUALS ____ M"
50 PRINT " A. 15"
60 PRINT " B. 1.5"
70 PRINT " C. 0.15"
80 PRINT "ENTER YOUR CHOICE (A, B OR C)";
90 INPUT ANS$
100 IF ANS$ = "B" THEN 130
110 PRINT "NO, THE ANSWER IS B"
120 GOTO 150
130 PRINT "THAT'S RIGHT!"
140 RIGHT = RIGHT + 1
150 PRINT
160 PRINT "(2) 3.5 KILOGRAMS EQUALS ____ GRAMS"
170 PRINT " A. 0.0035"
180 PRINT " B. 350"
190 PRINT " C. 3500"
200 PRINT "ENTER YOUR CHOICE (A, B OR C)";
210 INPUT ANS$
220 IF ANS$ = "C" THEN 250
230 PRINT "NO, THE ANSWER IS C"
240 GOTO 270
250 PRINT "THAT'S RIGHT!"
260 RIGHT = RIGHT + 1
270 PRINT
280 PRINT "YOU GOT"; RIGHT;" OUT OF 2 CORRECT."
290 END
```

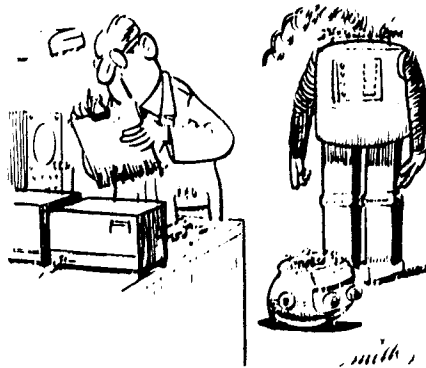
- (8) The following is a teacher's rough plans for writing a program to generate student report cards.

- (a) Input Student Name
- (b) Input Student Marks
- (c) GOSUB 1000 / Compute Average
- (d) GOSUB 2000 / Display Report Card
- (e) NEXT STUDENT

Write a program which will accomplish the teacher's objectives. Make it as general as possible.



## ARRAYS



**"GESUNDHEIT."**

---



TOPIC #12: ARRAYS

1. In BASIC, subscripted variables can be used to store more than one value under a given variable name. Another name for a subscripted variable is an array. All three types of variable (real, integer, and string) can be subscripted. Say you wish to store a person's name, address, and telephone number under the single variable name CLIENT\$. Then we could use:

```
CLIENT$(1) = "JOHN SMITH"
CLIENT$(2) = "1 BAKER STREET"
CLIENT$(3) = "382-4115"
```

The string variable CL\$ (the 'ient' may be omitted) has been assigned 3 separate storage locations. The client's name is stored in the first, his address in the second, and his phone number in the third.

DIM

2. Type in the following programme:

```
10 REM*MY AVERAGE MARK*
20 PRINT " ";TAB(16);"MARKS PROGRAMME"
30 PRINT TAB(16);" "
40 FOR I = 1 TO 1000:NEXT I
50 J = 1
60 ? " " " PLEASE ENTER A MARK";
70 INPUT MARK(J)
80 ?::?"DO YOU HAVE ANY MORE MARKS TO ENTER?"
85 ? "PRESS Y OR N. DO NOT PRESS 'RETURN'."
90 GET A$:IF A$ = ""GOTO 90
100 IF A$<>"Y" GOTO 120
110 J=J+1:GOTO 60
120 SUM = 0
130 FOR K = 1 TO J
140 : SUM = SUM + MARK(K)
150 NEXT K
160 AV = SUM/J
170 ? " " " YOUR AVERAGE MARK IS";AV
180 END
```

The programme will produce a chart such as the one below once all data has been entered (on a new screen):

| <u>FILL-UP NO.</u> | <u>DIST. (KM)</u> | <u>GAS (L.)</u> | <u>L./100 KM</u> |
|--------------------|-------------------|-----------------|------------------|
| 1                  | 342               | 29.4            | 8.6              |
| 2                  | 407               | 36.1            | 8.9              |
| ⋮                  | ⋮                 | ⋮               | ⋮                |

- b) At the end of the chart, print out the car's average fuel rating based on total distance and total gas consumed.
3. Write a BASIC programme that will allow you to type in two '2 by 2' matrices and print out their sum.

Example: MATRIX A =  $\begin{pmatrix} 5 & -2 \\ 4 & 3 \end{pmatrix}$  MATRIX B =  $\begin{pmatrix} 2 & 2 \\ 5 & -9 \end{pmatrix}$

their sum    MATRIX S    =  $\begin{pmatrix} 7 & 0 \\ 9 & -6 \end{pmatrix}$   $\begin{matrix} \leftarrow \text{row 1} \\ \leftarrow \text{row 2} \end{matrix}$   
column 1  $\uparrow$   $\uparrow$  column 2

Hint:  $S(I,J) = A(I,J)+B(I,J)$  where  $I$  is the row, and  $J$  is the column.

4. Write a BASIC programme that will allow you to type in two '2 by 2' matrices and print out their product. Hint: See #3 except that the product matrix P is determined by  $P(I,J) = A(I,1) * B(1,J) + A(I,2) * B(2,J)$ .

## EXERCISE

- (5) Write a program using two arrays that will:

- (a) Allow you to enter the name of cities and their populations,

```

Example: 10 PRINT "HOW MANY CITIES";
 20 INPUT N
 30 DIM CITY$(N), POP(N)
 | |
 | |
 array of array of
 names population sizes

```

- (b) Print out the cities having populations greater than 100 000.
- (c) Print out the number of such cities found.

- (6) Modify problem number 8 in previous exercise using arrays.

TOPIC #13: THE CASSETTE RECORDER

1. Make sure that the PET is turned off. Locate the cassette interface at the rear of the PET. It is the one furthest from the ON/OFF switch. Make sure that the 'slot' in the interface matches the 'divider' on the cassette connector. Gently slide the connector onto the interface. Now turn on the power. Never connect or disconnect a recorder when the computer is on. Power surges inside the micro can damage the computer!

EJECT

2. Press the EJECT lever on the cassette recorder. The tape drive door should open. Now take one of the demonstration cassettes and push it along the glide paths on the sides of the tape drive door until it clicks into place. The exposed tape should be facing out from the drive, and the description of the contents of the side of the tape you wish to use should appear in the window. Now push down the door until it snaps shut.

REWIND  
STOP  
LOAD

3. Press the REWIND lever on the cassette recorder. When the wheels stop turning, press the STOP lever. To enter the information stored on the tape into the PET's memory, type LOAD on the PET's keyboard and then press the RETURN key. The PET should display:

PRESS PLAY ON TAPE #1

PLAY

When you do press the PLAY lever, the PET will respond with

OK

and then in about 30 seconds,

FOUND NAME ..

LOADING

where NAME is the name of the first programme the PET has located on the tape. It will take a few minutes to load the programme into memory. Then

READY

☐

will be displayed. If the programme, found and loaded, was not the one you wanted, it may be that more than one programme is stored on that side of the tape. If this is the case, then type in LOAD once again and press the RETURN key. You will get the same type of display as above. Repeat this process until the programme you want has been found and loaded in.

4. Now press the STOP lever on the recorder and then the REWIND lever. When the wheel stops turning, press STOP again, and then press the EJECT lever to open the cassette door. Remove the cassette and close the door.
5. The PET is now ready to execute the programme. Type in RUN and press the RETURN key. To erase the stored programme, either type in NEW and press the RETURN key or simply turn the PET off.

6. If you know the exact 'name' of the programme you want to load, repeat step 3 but type in:

LOAD "NAME"

where NAME is the name of the programme. Notice that you must use double quotation marks. This procedure will allow the PET to keep searching until the correct programme has been located. Other programmes will be named as they are encountered.

7. If you have entered a programme, via the keyboard into the PET and wish to store it on tape for later use, follow this procedure:

- 1) Type in the following command and then press the RETURN key:

SAVE

SAVE "NAME"

where NAME is the name by which you wish to refer to the programme. Again notice the double quotes. The PET will respond by displaying:

PRESS PLAY AND RECORD ON TAPE #1

RECORD

- 2) Enter a blank tape in the recorder then press both the PLAY and RECORD levers at the same time. The PET will respond by displaying: OK

WRITING NAME

- 3) After a few minutes the recorder will stop and the PET will display: READY

Now rewind the tape using the REWIND lever and then the STOP lever.

VERIFY

- 4) Next, type in VERIFY "NAME" and press the RETURN key. The PET will now display:

PRESS PLAY ON TAPE #1

- 5) Press PLAY lever on the recorder, and the PET will display:

SEARCHING FOR "NAME"

FOUND "NAME"

VERIFYING

OK

READY

⏏

This will take a few seconds.

- 6) Finally rewind the tape and remove it from the recorder. Notice that now the programme is stored both in the PET (until it is turned off or altered) and on tape (until it is erased or copied over).
  - 7) If you get a VERIFICATION ERROR message, repeat the whole saving process from the start.
- 

#### REVIEW EXERCISE

1. Take some of the 'canned' programmes that are available on tape and practice loading them into the PET. Try running each one after it has been entered.
2. Type in a simple BASIC programme (for example, see section 2 Topic 3). Take a blank cassette and try to 'save' your programme on tape. Turn the PET off to erase your programme and then try to re-load it from the tape.



TOPIC #14: THE PRINTER

1. To connect the printer to your PET, follow these steps:
  - 1) Make sure that neither the PET nor the printer are connected to AC power (unplug them).
  - 2) Plug the small end of the connector cable into the interface slot in the back of the printer and secure it by tightening the two screws.
  - 3) Plug the other end into the interface nearest to the on/off switch at the back of the PET. Make sure that the connector side with the word 'COMMODORE' on it is facing up.
  - 4) Plug in both the PET and the printer. Locate the rocker switch at the back of each and turn them on.
2. To open the communication line from the PET to the printer you must use an OPEN statement. Type:

OPEN

OPEN 1,4

The 1 in this statement can be any integer from 1 to 255. The 4 refers to the printer. To have single lines of output printed by the printer use the PRINT# statement. Type:

PRINT #1, "MY FIRST PRINTOUT"

The ? cannot be used here instead of PRINT. The number 1 in this statement can be any number from 1 to 255 but must match the one chosen in the OPEN statement. The communication line between the PET and the printer should be closed before moving on to the next programme. To do this type:

CLOSE 1

and press the RETURN key.

3. Type in a simple programme on the keyboard of the PET (Use any one of the programmes you have previously written). To have your programme print out on the printer, instead of on the screen, replace the PRINT's or ?'s with PRINT#1,'s. For example:

|                    |           |                           |
|--------------------|-----------|---------------------------|
| 10 REM*EXAMPLE*    | } becomes | 10 REM*EXAMPLE*           |
| 20 FOR I = 1 TO 10 |           | 15 OPEN 1,4               |
| 30 ? "I LOVE PET"; |           | 20 FOR I = 1 TO 10        |
| 40 NEXT I          |           | 30 PRINT#1, "I LOVE PET"; |
| 50 END             |           | 44 NEXT I                 |
|                    |           | 45 CLOSE 1                |
|                    |           | 50 END                    |

Notice the comma after the #1. Now run the programme and notice where the output appears.

CMD

4. A permanent listing of any programme can be obtained on the printer using the CMD statement. This CMD (command) statement transfers print command to the device specified. If you used OPEN 1,4 then use CMD 1 to transfer command to the printer. Now anything that would normally be displayed on the screen will instead be printed by the printer. When LIST is now used, the programme will be listed by the printer instead of being displayed on the screen, as before.

Type:

OPEN 1,4

CMD 1

LIST

When the RETURN key is pressed the listing will be printed out by the printer.

CLOSE

5. After every use, the printer must be closed using the CLOSE command. Between every OPEN and CLOSE statement must appear at least one PRINT# statement. A PRINT# must occur between the CMD statement and the CLOSE command. Type:

PRINT#1,"GOODBYE"

CLOSE 1

---

#### REVIEW QUESTIONS

1. Use the printer to list some of the programmes you have written in Basic.
2. Load the PET with a programme selected from the 'canned' programmes and get a listing on the printer.



EXTRA NOTES

A. PET BASIC RESERVED WORDS (see section 7, topic 3)

|       |        |         |        |
|-------|--------|---------|--------|
| ABS   | GET    | OPEN    | SPC    |
| AND   | GET#   | OR      | SQR    |
| ASC   | GOSUB  | PEEK    | ST     |
| ATN   | GOTO   | POKE    | STEP   |
| CHR\$ | IF     | POS     | STOP   |
| CLOSE | INPUT  | PRINT   | STR\$  |
| CLR   | INT    | PRINT#  | SYS    |
| CMD   | LEFT\$ | READ    | TAB    |
| CONT  | LEN    | READ#   | TAN    |
| COS   | LET    | REM     | THEN   |
| DATA  | LIST   | RESTORE | TI     |
| DEF   | LOAD   | RETURN  | TI\$   |
| DIM   | LOG    | RIGHT\$ | TO     |
| END   | MID\$  | RND     | USR    |
| EXP   | NEW    | RUN     | VAL    |
| FN    | NEXT   | SAVE    | VERIFY |
| FOR   | NOT    | SGN     | WAIT   |
| FRE   | ON     | SIN     |        |

B. ALTERNATE CHARACTER SET

- 1) To activate the alternate character set, either type  
POKE 59468,14  
directly, or use it as a statement in your programme.
- 2) To re-activate the standard character set, either type  
POKE 59468,12  
directly, or use it as a statement in your programme.
- 3) The alternate character set changes the keyboard so that upper and lower case characters are available for each alphabetic key. The graphic symbols are thus not available.

C. USER DEFINED FUNCTIONS

DEFFN

In PET BASIC it is possible to create your own 'built-in' functions. The DEFFN (define function) statement is used. The function name, which must follow DEFFN can be any valid real number variable name. The following example defines the function  $g(X) = X^2$  and uses it to calculate  $5X^2 - 3$ :

```
10 DEF FN G(X) = X↑2
```

```
20 A = 7
```

```
30 ? A, 5*FNG(A) - 3
```

Notice that once the function is defined, in line 10, any real variable name can be used in the argument (As long as it has been assigned a numeric value).

D. ENHANCED CHARACTER PRINTOUT

CHR\$(1)

When using the printer it is possible to double the width of the characters printed, by using the CHR\$(1) code. All characters following CHR\$(1) will be printed double width. A second CHR\$(1) in the same PRINT# statement will double the width again to 4 times its normal size. A third will double it again, and so on.

examples: 10 PRINT#1, "FUN";CHR\$(1)"WOW!"

```
20 PRINT#1, "BO";CHR$(1)"O";CHR$(1)"O"
```

```
30 PRINT#1, CHR$(1)"WIDE";CHR$(129)"THIN"
```

Notice that CHR\$(129) can be used to cancel this process, and that commas must not be used to separate CHR\$(1) from the strings that follow it.